**Programs:**

**Java MCQ / programs**

1. [**https://www.geeksforgeeks.org/java-tricky-output-questions/?ref=lbp**](https://www.geeksforgeeks.org/java-tricky-output-questions/?ref=lbp)

**1. convert hashmap into ArrayList**

**class Test{**

**public static void main(String[] args) {**

**HashMap<Integer, String> customerIdNameMap = new HashMap<Integer, String>();**

**// Putting key-values pairs in HashMap**

**customerIdNameMap.put(1001, "Arman");**

**customerIdNameMap.put(1002, "Javin");**

**customerIdNameMap.put(1003, "Mat");**

**customerIdNameMap.put(1004, "Joe");**

**// Before Java 8**

**// Convert keys to ArrayList**

**Set<Integer> keySet = customerIdNameMap.keySet();**

**List<Integer> customerIdList = new ArrayList<Integer>(keySet);**

**System.*out*.println("customerIds: "+customerIdList);**

**// Convert values to ArrayList**

**Collection<String> values = customerIdNameMap.values();**

**List<String> customerNames = new ArrayList<String>(values);**

**System.*out*.println("Customer Names: "+ customerNames);**

**// Convert entry objects to ArrayList**

**Set<Entry<Integer, String>> entrySet = customerIdNameMap.entrySet();**

**List<Entry<Integer,String>> entryCustomerList = new ArrayList<Map.Entry<Integer,String>>(entrySet);**

**System.*out*.println("Customer ID and Names: "+entryCustomerList);**

**}**

**}**

1. **convert hashmap into ArrayList using java 8**

**class Test {**

**public static void main(String[] args) {**

**HashMap<Integer, String> customerIdNameMap = new HashMap<Integer, String>();**

**// Putting key-values pairs in HashMap**

**customerIdNameMap.put(1001, "Arman");**

**customerIdNameMap.put(1002, "Javin");**

**customerIdNameMap.put(1003, "Mat");**

**customerIdNameMap.put(1004, "Joe");**

**// Java 8**

**// Convert keys to ArrayList**

**List<Integer> customerIdList = customerIdNameMap.keySet().stream().collect(Collectors.*toList*());**

**System.*out*.println("customerIds: " + customerIdList);**

**// Convert values to ArrayList**

**List<String> customerNames = customerIdNameMap.values().stream().collect(Collectors.*toList*());**

**System.*out*.println("Customer Names: " + customerNames);**

**// Convert entry objects to ArrayList**

**List<Entry<Integer, String>> entryCustomerList = customerIdNameMap.entrySet().stream()**

**.collect(Collectors.*toList*());**

**System.*out*.println("Customer ID and Names: " + entryCustomerList);**

**}**

**}**

**3.find out integer start with 1 using streams**

**public class Demo {**

**public static void main(String[] args) {**

**List<Integer> list = Arrays.*asList*(1, 10, 20, 30, 23, 89, 100, 1000, 120);**

**List<Integer> result = list.stream().map(String::*valueOf*).filter(f -> f.startsWith("1")).map(Integer::*parseInt*)**

**.collect(Collectors.*toList*());**

**System.*out*.println(result);**

**}**

**}**

1. **using java 8 streams filter the employees more than 50000 salary**

**class Employee {**

**private String name;**

**private Integer salary;**

**private String dept;**

**public Employee(String name, Integer salary, String dept) {**

**this.name = name;**

**this.salary = salary;**

**this.dept = dept;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public Integer getSalary() {**

**return salary;**

**}**

**public void setSalary(Integer salary) {**

**this.salary = salary;**

**}**

**public String getDept() {**

**return dept;**

**}**

**public void setDept(String dept) {**

**this.dept = dept;**

**}**

**@Override**

**public String toString() {**

**return "Name: " + name + ", Salary: " + salary + ", Dept: " + dept;**

**}**

**@Override**

**public int hashCode() {**

**return Objects.*hash*(name);**

**}**

**@Override**

**public boolean equals(Object obj) {**

**if (this == obj)**

**return true;**

**if (obj == null)**

**return false;**

**if (getClass() != obj.getClass())**

**return false;**

**Employee other = (Employee) obj;**

**return Objects.*equals*(name, other.name);**

**}**

**}**

**public class Test {**

**public static void main(String[] args) {**

**List<Employee> emp = new ArrayList<>();**

**emp.add(new Employee("emp1", 10000, "HR"));**

**emp.add(new Employee("emp2", 60000, "HR"));**

**emp.add(new Employee("emp3", 70000, "Development"));**

**emp.add(new Employee("emp4", 100000, "Developement"));**

**emp.add(new Employee("emp5", 8000, "Testing"));**

**List<Employee> empR = emp.stream().filter(i -> i.getSalary() > 50000).collect(Collectors.*toList*());**

**empR.forEach(System.*out*::println);**

**}**

**}**

1. **how to group the employees by department**

**public class Test {**

**public static void main(String[] args) {**

**List<Employee> emp = new ArrayList<>();**

**emp.add(new Employee("emp1", 10000, "HR"));**

**emp.add(new Employee("emp2", 60000, "HR"));**

**emp.add(new Employee("emp3", 70000, "Development"));**

**emp.add(new Employee("emp4", 100000, "Development"));**

**emp.add(new Employee("emp5", 8000, "Testing"));**

**Map<String,List<Employee>> empR = new HashMap<>();**

**// before Java8**

**List<Employee> value=new ArrayList<>();**

**for(Employee e : emp) {**

**String key = e.getDept();**

**if(!empR.containsKey(e.getDept())) {**

**empR.put(key, new ArrayList<>());**

**}**

**empR.get(key).add(e);**

**}**

**empR.entrySet().stream().forEach(System.*out*::println);**

**//Java8 version**

**empR = emp.stream().collect(Collectors.*groupingBy*(Employee :: getDept));**

**System.*out*.println("====Java 8 version output====");**

**empR.entrySet().stream().forEach(System.*out*::println);**

**}**

**}**

1. **write the program to find the occurrence of each character in string**

**class Test {**

**public static void main(String[] args) {**

**String s = "Programming";**

**String[] c = s.split("");**

**// in Java 7**

**Map<String, Integer> m = new HashMap<String, Integer>();**

**for (int i = 0; i < c.length; i++) {**

**String key = c[i];**

**if (m.containsKey(key))**

**m.put(key, m.get(key) + 1);**

**else**

**m.put(key, 1);**

**}**

**// in java 8**

**List<String> str = new ArrayList<String>(Arrays.*asList*(c));**

**Map<String, Integer> map = new HashMap<String, Integer>();**

**str.stream().forEach(chr -> {**

**if (map.containsKey(chr))**

**map.put(chr, map.get(chr) + 1);**

**else**

**map.put(chr, 1);**

**});**

**System.*out*.println("in java 7 output====");**

**m.entrySet().stream().forEach(System.*out*::println);**

**System.*out*.println("in java 8 output====");**

**map.entrySet().stream().forEach(System.*out*::println);**

**}**

**}**

1. **write the program to find the max occurrence of character in string**

**class Test {**

**public static void main(String[] args) {**

**String s = "Programming";**

**String[] c = s.split("");**

**// in java 8**

**List<String> str = new ArrayList<String>(Arrays.*asList*(c));**

**Map<String, Integer> map = new HashMap<String, Integer>();**

**str.stream().forEach(chr -> {**

**if (map.containsKey(chr))**

**map.put(chr, map.get(chr) + 1);**

**else**

**map.put(chr, 1);**

**});**

**Set<String> s1 = new HashSet<>();**

**Integer value = map.values().stream().max(Integer:: *compare*).get();**

**map.entrySet().stream().forEach(entry->{**

**if(value==entry.getValue())**

**s1.add(entry.getKey());**

**});**

**s1.stream().forEach(System.*out*::println);**

**}**

**}**

1. **write a program to find number of integer occurrence in array, if frequency repeated return max number of array**

**//Main class**

**class Test {**

**public static void main(String[] args) {**

**Integer[] intArray = { 10, 20, 20, 20, 30, 30, 30, 40 };**

**// in java 8**

**List<Integer> intList = new ArrayList<>(Arrays.*asList*(intArray));**

**Map<Integer, Integer> map = new HashMap<Integer, Integer>();**

**intList.stream().forEach(chr -> {**

**if (map.containsKey(chr))**

**map.put(chr, map.get(chr) + 1);**

**else**

**map.put(chr, 1);**

**});**

**Set<Integer> s1 = new HashSet<>();**

**Integer value = map.values().stream().max(Integer::*compare*).get();**

**map.entrySet().stream().forEach(entry -> {**

**if (value == entry.getValue())**

**s1.add(entry.getKey());**

**});**

**if (s1.size() == 1) {**

**System.*out*.println("Maximum occuring integer is===");**

**s1.stream().forEach(System.*out*::println);**

**} else {**

**Integer max = map.keySet().stream().max(Integer::*compare*).get();**

**System.*out*.println("Maximum integer is===" + max);**

**}**

**}**

**}**

1. **find the output of below program**

**package com.main.test;**

**//Main class**

**class Test {**

**public static void gfg(String s)**

**{**

**System.*out*.println("String");**

**}**

**public static void gfg(Object o)**

**{**

**System.*out*.println("Object");**

**}**

**public static void main(String[] args) {**

**System.*out*.println(100+200+"Java"); //300java**

**System.*out*.println("java"+100+200); //java100200**

**System.*out*.println('j'+'a'+'v'+'a'); //418**

***gfg*(null); //String**

**}**

**}**

* **In case of**[**method overloading**](https://www.geeksforgeeks.org/overloading-in-java/)**, the most specific method is chosen at compile time.**
* **As ‘java.lang.String’ is a more specific type than ‘java.lang.Object’.**
* **In this case the method which takes ‘String’ as a parameter is chosen.**
* **If we have one more method which accept Integer object, As ‘java.lang.String’ and ‘java.lang.Integer’ is a more specific type than ‘java.lang.Object’,but between ‘java.lang.String’ and ‘java.lang.Integer’ none is more specific.**
* **In this case the Java is unable to decide which method to call. So compiler error will throw.**

1. **Write the program to find the first non repeated char from the string**

**public class FirstNonRepeatedCharFirst {**

**public static void main(String args[]) {**

**String inputStr ="teeter";**

**for(char i :inputStr.toCharArray()){**

**if ( inputStr.indexOf(i) == inputStr.lastIndexOf(i)) {**

**System.out.println("First non-repeating character is: "+i);**

**break;**

**}**

**}**

**}**

**}**

1. **Write the program to split the number as integer and multiply untill single digit like below**

**39 → (3x9=) 27 →**

**Step 2(2x7=) 14 →**

**Step 3(1x4=) 4**

1. **Sort the names in descending order & find the sum of integer squares**

**class Test {**

**// Bhuvaneswari,Sundar, Komal**

**// Sundar,Komal,Bhuvanesari**

**List<String> names = new ArrayList<>();**

**names.add("Bhuvana");**

**names.add("Sundar");**

**names.add("Komal");**

**names.stream().sorted((n1,n2)->n2.compareTo(n1)).forEach(System.*out*::println);**

**Integer var = Stream.*of*(2,4,6,8).max(Integer::*compare*).get();**

**System.*out*.println("=====max value==="+var);**

**Integer min = Stream.*of*(2,4,6,8).min(Integer::*compare*).get();**

**System.*out*.println("=====min value==="+min);**

**//2,4,6,8**

**//Sum of Squrs**

**IntSummaryStatistics sum = Stream.*of*(2,4,6,8).map(i->i\*i).collect(Collectors.*summarizingInt*(Integer::intValue));**

**System.*out*.println("=====max value==="+sum.getMax());**

**System.*out*.println("=====min value==="+sum.getMin());**

**System.*out*.println("=====sum value==="+sum.getSum());**

**System.*out*.println("=====count value==="+sum.getCount());**

**}**

**}**

1. **Write a program to find given two strings are anagram or not**

**public boolean getAnagram(String s1, String s2){**

**Character[] c1 = s1.toCharArray();**

**Character[] c2 = s2.toCharArray();**

**Arrays.sort(c1);**

**Arrays.sort(c2);**

**if(c1.length == c2.length){**

**/\*for(int i=0;i<c1.length;i++){**

**if(c1[i]!=c2[i])**

**return false;**

**}**

**return true;**

**\*/**

**Return Arrays.equals(c1,c2);**

**}else**

**return false;**

**}**

1. **Write a program to distribute T number of candies to n number of students based on scores**
2. **Remove duplicates in the list using stream**

**List<Integer> newList = list.stream()**

**.distinct()**

**.collect(Collectors.toList());**

1. **Find the subfolder from given array whose sum Is given sum**

**package com.example.interview;**

**import java.util.ArrayList;**

**import java.util.Collection;**

**import java.util.Collections;**

**import java.util.HashMap;**

**import java.util.List;**

**import java.util.Map;**

**public class Test {**

**public static void main(String[] args) {**

**Integer arr[] = {1, 4, 20, 3, 10, 5};**

**Integer sum = 33;**

**Integer num=0;**

**Integer start=0;**

**Integer end=0;**

**for(int i =0;i<arr.length;i++) {**

**num=arr[i];**

**start=i;**

**for(int j =i+1;j<arr.length;j++) {**

**num=num+arr[j];**

**if(num>sum)**

**break;**

**if(num==sum) {**

**end=j;**

**System.out.println("====start=="+start+"===end===");**

**display(start,end,arr);**

**}**

**}**

**}**

**}**

**public static void display(Integer start,Integer end,Integer arr[] ) {**

**for(int i=start;i<=end;i++) {**

**System.out.println(arr[i]);**

**}**

**}**

**}**

1. **Find the second largest number from array using stream**

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Integer arr[] = { 20, 1, 40, 20, 3, 10, 5, 40, 20 };

List<Integer> list = Arrays.*asList*(arr);

Integer max1 = list.stream().max(Integer::*compare*).get();

Integer num = list.stream().filter(i -> i != max1).max(Integer::compareTo).get();

System.***out***.println("====max value using two lines===" + num);

Integer max = list.stream().sorted(Collections.*reverseOrder*()).distinct().limit(2).skip(1).findFirst().get();

System.***out***.println("====max value===" + max);

System.***out***.println(list.stream().sorted(Collections.*reverseOrder*()).collect(Collectors.*toList*()));

// [40, 40, 20, 20, 20, 10, 5, 3, 1]

System.***out***.println(list.stream().sorted(Collections.*reverseOrder*()).distinct().collect(Collectors.*toList*()));x

//[40, 20, 10, 5, 3, 1]

System.***out***.println(

list.stream().sorted(Collections.*reverseOrder*()).distinct().limit(2).collect(Collectors.*toList*()));

//[40, 20]

System.***out***.println(list.stream().sorted(Collections.*reverseOrder*()).distinct().limit(2).skip(1)

.collect(Collectors.*toList*()));

//[20]

}

}

1. **What Is the output of below program**

int test() {

  try {

    return 1;

  }

  finally {

    return 2;

    SOP("finally");

  }

  return 3;

}

**it will throw unreachable code compilation error on SOP(finally) and return 3 statement. Because in finally return statement is there. If we removed those statement the output is return 2.**

1. **Output of below program**

**class** A {

**void** m1() {

System.***out***.println("class A");

}

}

**class** B **extends** A {

**void** m1() {

System.***out***.println("class B");

}

}

**public** **class** Test **extends** B {

**void** m1() {

System.***out***.println("class C");

}

**public** **static** **void** main(String args[]) {

A a=**new** B();

Test c=(Test)a;

c.m1();

}

}

* It will throw below run time exception
* Exception in thread "main" java.lang.ClassCastException: class com.example.interview.B cannot be cast to class com.example.interview.Test (com.example.interview.B and com.example.interview.Test are in unnamed module of loader 'app')

at com.example.interview.Test.main(Test.java:33)

* **Because child reference cannot used to hold parent object**

1. **How to find the magic number**
2. **How to sort the HashMap based on keys**

**Sql/Query**

1. **write a query to find the second maximum salary of employee**

* **select name, max(salary) from Employee where salary not in (select max(salary) from employee) (need to test in local)**
* **select min(salary) from  (select distinct salary from emp order by salary desc)  where rownum < 3;**

1. **write a query to get the login status of user based on timestamp (last login status)**
2. **select dept , sum of salary from each department**
3. **find the min , max salary of given employee**
4. **what is table & field**
5. **groupby, max,min, limit sql quries**
6. **what is index**

**Design Pattern**

1. **What is design pattern?**
2. **what are the design pattern you worked?**
3. **what is singleton design pattern, Write the program for Singleton Design Pattern**
4. **How to break singleton design pattern**
5. **what is Factory Design pattern, Write the program for Factory Design Pattern**

**Jdbc/Hibernate/JPA**

**1.What is the JDBC?**

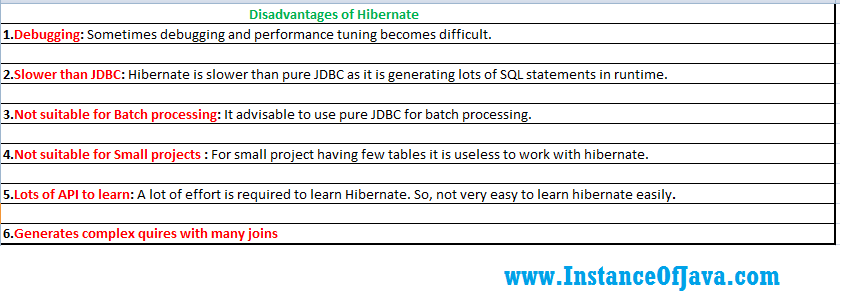
* **JDBC stands for Java Database Connectivity.**
* **It is a free open-source application programming interface for Java which enables the application to access the database.**
* **It enables the developers to create queries, update data to the relational database using a Structured Query Language (SQL).**

**2. what is hibernate? why we are using hibernate, advantage, disadvantage**

* **Hibernate is an open-source and lightweight ORM tool that is used to store, manipulate, and retrieve data from the database.**
* **ORM is an acronym for Object/Relational mapping. It is a programming strategy to map object with the data stored in the database. It simplifies data creation, data manipulation, and data access.**

**Advantage:**

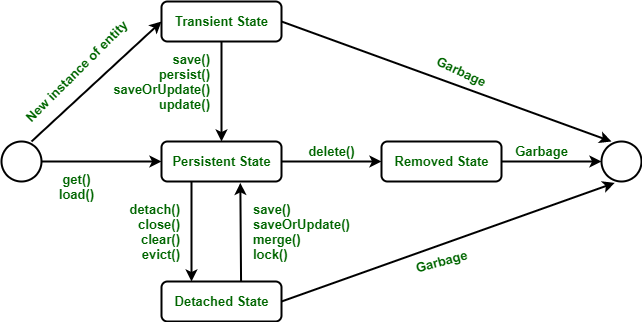
* **It is ORM tool**
* **Hibernate has an exception translator, which converts checked exceptions of JDBC in to unchecked exceptions of hibernate. So all exceptions in hibernate are unchecked exceptions and Because of this no need to handle exceptions explicitly.**
* **Hibernate has its own query language called Hibernate Query Language. With this HQL hibernate became database independent.**
* **Hibernate supports relationships like One-To-One, One-To-Many, Many-To-One ,Many-To-Many.**
* **Hibernate has Caching mechanism. using this number of database hits will be reduced. so performance of an application will be increases.**
* **Hibernate also supports annotations along with XML.**
* **Hibernate supports Lazy loading.**
* **Hibernate is easy to learn it is developers friendly.**
* **Hibernate maintains database connection pool.**
* **Hibernate has Concurrency support.**

[**[](https://1.bp.blogspot.com/-zGgLuXdqU8U/WAtr0-Y_TeI/AAAAAAAABVg/4hKLy7M8AQcRXslZFSEytlV38YGdnfpYQCLcB/s1600/advantages%2Band%2Bdisadvantages%2Bof%2Bhibernate.png)**](https://1.bp.blogspot.com/-zGgLuXdqU8U/WAtr0-Y_TeI/AAAAAAAABVg/4hKLy7M8AQcRXslZFSEytlV38YGdnfpYQCLcB/s1600/advantages%2Band%2Bdisadvantages%2Bof%2Bhibernate.png)

1. **Hibernate life cycle?**

**There are mainly four states of the Hibernate Lifecycle :**

1. **Transient State**
2. **Persistent State**
3. **Detached State**
4. **Removed State**

****

**Eg:** [**https://www.geeksforgeeks.org/hibernate-lifecycle/#:~:text=There%20are%20mainly%20four%20states,Detached%20State**](https://www.geeksforgeeks.org/hibernate-lifecycle/#:~:text=There%20are%20mainly%20four%20states,Detached%20State)

**4. Difference between first level and second level cache**

| **Sr. No.** | **Key** | **First level cache** | **Second level cache** |
| --- | --- | --- | --- |
| **1** | **Basic** | **First level cache is a session level cache and it is always associated with session level object** | **Second level cache is session factory level cache and it is available across all sessions** |
| **2** | **Enabled** | **It is enabled by default.** | **It is not enabled by default.** |
| **3** | **Availability** | **It is available for a session** | **It is available across all session.** |
| **4** | **Configuration** | **No Extra configuration required** | **We have to decide  which concurrency strategy to use and also need to configure cache expiration and physical cache attributes.** |

**5.differennce between update and merge?**

| **Update** | **Merge** |
| --- | --- |
| **Hibernate proprietary method** | **Part of the JPA specification** |
| **Does not return anything, returns a void** | **Returns the updated object** |
| **In update() method, if the object already exists in Cache, then update() method will throw an Exception.** | **In merge() method, if the object already exists in Cache, then merge() method will simply copy the object into Cache.** |

**6.difference between get and load in hibernate?**

| **Sr. No.** | **Key** | **Get()** | **Load()** |
| --- | --- | --- | --- |
| **1** | **Basic** | **It is used to fetch data from the database for the given identifier** | **It  is also used to fetch data from the database for the given identifier** |
| **2** | **Null Object** | **It object not found for the given identifier then it will return null object** | **It will throw object not found exception** |
| **3** | **Lazy or Eager loading** | **It hit the database and returns original object** | **It always returns proxy object** |
| **4** | **Performance** | **It is slower than load() because it return fully initialized object which impact the performance of the application** | **It is slightly faster.** |
| **5.** | **Use Case** | **If you are not sure that object exist then use get() method** | **If you are sure that object exist then use load() method** |

**7. hibernate criteria**

**8.optimistic locking in hibernate?**

**9.what are the inheritance mapping strategies?**

**10. hibernate mapping with annotations**

**11.How do you create Relations between the entity**

**12.Write a program to define relationship between the Employee and department, 1 dept have multiple employees**

[**https://www.baeldung.com/hibernate-one-to-many**](https://www.baeldung.com/hibernate-one-to-many)

**https://www.javatpoint.com/hibernate-one-to-many-mapping-using-annotation-example**

**11.statement vs prep statement?**

**what is jpa and use of jpa?**

**What is the implementation of JPA**

**11. Difference between Hibernate and Spring jpa?**

**12. did jpa is perform actual task?**

**No implementation like hibernate will perform actual task.**

**Spring Boot/Spring/rest**

**1.Difference between Spring and Spring boot / which one you prefer spring or spring boot?**

|  |  |
| --- | --- |
| **Spring** | **Spring Boot** |
| **Spring Framework is a widely used Java EE framework for building applications.** | **Spring Boot Framework is widely used to**  **develop REST APIs.** |
| **It aims to simplify Java EE development that makes developers more productive.** | **It aims to shorten the code length and provide the easiest way to develop Web Applications.** |
| **The primary feature of the Spring Framework is dependency injection.** | **The primary feature of Spring Boot is Autoconfiguration. It automatically configures the classes based on the requirement.** |
| **It helps to make things simpler by allowing us to develop loosely coupled applications.** | **It helps to create a stand-alone application with less configuration.** |
| **The developer writes a lot of code (boilerplate code) to do the minimal task.** | **It reduces boilerplate code.** |
| **To test the Spring project, we need to set up the sever explicitly.** | **Spring Boot offers embedded server such as Jetty and Tomcat, etc.** |
| **It does not provide support for an in-memory database.** | **It offers several plugins for working with an embedded and in-memory database such as H2.** |
| **Developers manually define dependencies for the Spring project in pom.xml.** | **Spring Boot comes with the concept of starter in pom.xml file that internally takes care of downloading the dependencies JARs based on Spring Boot Requirement.** |

**2.Difference between controller and rest controller?**

* **The @RestController annotation in Spring MVC is nothing but a combination of the @Controller and the @ResponseBody annotation.**
* **The @Controller annotation indicates that a particular class serves the role of a controller.**
* **The @ResponseBody annotation tells a controller that the object returned is automatically serialized into JSON and passed back into the HttpResponse object.**

**@Controller**

**@ResponseBody**

**public class MVCController { }**

**@RestController**

**public class MVCController { }**

**3. what are the annotation used in spring boot**

1. **@SpringBootApplication**

* **It is combination of @EnableAutoConfiguration , @ComponentScan,@Configuration,**

1. **@enableAutoConfiguration**

* **Automatically configure an application based on the dependencies present in the classpath**

1. **@ComponentScan**

* **When we use @ComponentScan explicitly anywhere in the application, by default spring boot application scans all the beans (using @Component annotation) under the base package**
* **Without this annotation spring boot could not recognize the beans and we can’t add them into application context and inject the dependency**

1. **@Configuration**

* **On startup spring creates the application context . this is the object that stores all these dependencies so they can be wired into the application where they are needed**
* **Spring refers these dependencies as the beans**
* **Annotating a class with the @Configuration indicates that the class can be used by the Spring IoC container as a source of bean definitions.**

1. **@Bean**

**The @Bean annotation tells Spring that a method annotated with @Bean will return an object that should be registered as a bean in the Spring application context.**

1. **@RestController**

* **this annotation responsible for handling all the incoming request and map the url request to the service methods**
* **it is the combination of @Controller and @ResponseBody(both required in every controller class)**

1. **@Component - this is used to add/register the object/bean to the application context**
2. **@Autowired**

* **enable you to inject object dependency implicitly**
* **it is internally using setter or constructor injection**
* **it can’t use to inject string and primitive values .it works with reference only**

1. **@Value**

* **It is used to Inject property values(appliction.property) into the component**
* **Basically it is inject all the values from environment variables or spring cloud config**

[**9.@Controller**](mailto:9.@Controller)**, @Service,@RequestMapping, @GetMapping, @PostMapping,@PutMapping@DeleteMapping**

**4. Purpose of controller class.**

* **Controller class used to handle the client request.**
* **It will receive the client request and invoke business class to perform business related activities.**
* **And return the response to client,**
* **It is achieved using @RestController/ @Controller with @ResponseBody**

**5. types of Request mapping?**

* **@RequestMapping *annotation is used to map web requests to Controller methods.***

**Types:**

1. **@GetMapping – used to get**
2. **@PostMapping – used to post**
3. **@PutMapping – used to update**
4. **@DeleteMapping – used to delete**
5. **@PatchMapping – used to update**

**6. difference between request mapping and get mapping**

|  |  |
| --- | --- |
| **Request mapping** | **Get Mapping** |
| **It is used in class and method level** | **It is used with method level** |
| **It is used to handle all the Http request** | **It is used to handle only Http Get methods** |
| **@RequestMapping(method = RequestMethod.GET)** | **@GetMapping** |

**7. what is dependency injection in spring and types of dependency injection.**

* **Class may contain different dependencies, to create object of the class we need initialize dependencies also**
* **Instead of programmer spring is creating, managing the objects**
* **Spring injecting objects through constructor or setter method**
* **Object type dependencies injected using @Autowire**
* **Literals are injected through @value**

**8.what is Autowiring / what is @Autowired**

* **Autowiring means, we can wire/inject the object anywhere in the application**
* **It is done using @Autowired**
* **It can be done for object type only, not for string, primitive types**
* **It can be done in constructor/setter method level**

**9. what is @EnableAutoConfigure**

* **Spring Boot auto-configuration attempts to automatically configure your Spring application based on the jar dependencies that you have added.**
* **For example, If HSQLDB is on your classpath, and you have not manually configured any database connection beans, then we will auto-configure an in-memory database.**
* **You need to opt-in to auto-configuration by adding the @EnableAutoConfiguration or @SpringBootApplication annotations to one of your @Configuration classes.**
* **To exclude auto configuration for data source**
* ***@EneableAutoConfiguration(exclude={DataSourceAutoConfiguration.class})***

**10.use of @Qualifier**

* **The @Qualifier annotation is used to resolve the autowiring conflict, when there are multiple beans of same type.**
* **The @Qualifier annotation can be used on any class annotated with @Component or on method annotated with @Bean. This annotation can also be applied on constructor arguments or method parameters.**

[**https://www.boraji.com/spring-4-qualifier-annotation-example**](https://www.boraji.com/spring-4-qualifier-annotation-example)

**11.run spring boot application in external server**

[**https://dzone.com/articles/spring-boot-with-external-tomcat**](https://dzone.com/articles/spring-boot-with-external-tomcat)

**12.What is @Profile, how to create profile? What is default profile? (video required)**

* **Spring boot profiles provides the way to segregate application configuration**
* **It makes useful in configuring different environment like dev,prod,qa**

**Create Profile:**

* + **We can create profile under resources, eg : for dev env profile is application-dev.properties**
  + **Also we can create profile using java like below**
  + **Any @Component or @Configuration can be marked with @Profile to limit when it is loaded:**

***@Configuration***

***@Profile("production")***

**public class ProductionConfiguration {**

***// ...***

**}**

* **In application.properties file add spring.profile.active=dev**
* **Or else we can set the profile in command prompt**

**13.springboot starters?**

* **Spring boot has built-in starters which makes our development easier**
* **These starters adding relevant jars in our class path**
* **For example if we want to use Spring and JPA for database access, we need to include the spring-boot-starter-data-jpa dependency in our build tool file of the project.**

**Starters used in our project:**

* + **org.springframework.boot:spring-boot-starter-web – includes restful, tomcat**
  + **spring-boot-starter-data-jpa – for jpa data access**
  + **org.springframework.boot:spring-boot-starter-actuator – for health chek**
  + **junit**
  + **echace**
  + **swagger**
  + **hibernate**

**14. what is actuators?**

1. **is the mechanism where we can monitor the health of our applications in production .**
2. **like how many sessions, connections, state of database resources and if we want to analyze the log**
3. **monitor and manage your application when you push it to production**

**15. what is spring boot scopes – need to study full bean scopes (video)**

* **Singleton - Singleton is the default scope for a Bean, the one that will be used if nothing else is indicated. This scope implies that Spring container will create an only shared instance of the class designated by this bean, so each time the Bean is required the same object will be injected.**
* **Prototype**
* **Request**
* **Session**
* **Global session**
* **Application**
* **Websocket**

**16.what is default scope in spring boot and behavior (eager, lazy)? how to change it using annotation?**

* **By default it is singleton**
* **And lazy loading when we are using @Autowire that time only bean will be created**

**17.how to change default server tomcat to jetty in spring boot?**

**18.what is lambok?**

**19.how application.properties will be read by spring boot application?**

**20.what are the framework used in spring boot for database operation, where you will mention the connection details In spring boot application.**

* **Jpa and hibernate for the implementation**
* **In application.properties or cloud config**

**21.how the connection happens to the data base in spring boot applications. what are the annotations required to create connections**

**22.how to configure multiple db using spring boot. How spring boot knows which one needs to confgure first**

* **In cloud config we can declare multiple data source**
* **In our configuration file we have to create bean for those data source**
* **By maiking it one bean as @primary it will configure first.**
* **Using bean name and @Qualifier annotation we can configure second datasource connection**

**23.how the dependency jars/libraries added into spring boot project**

**24.how to configure spring beans**

**25.what is spring MVC**

**26.what is spring security**

* **Spring boot provides the default form based authentication to authenticate the users**
* **To enable this feature we need to add Spring-boot-starter-security starter in out build tool , it will add the dependencies required for spring security.**
* **Whereas user name is user and password is randomly generated password. Password found in console.**
* **Most of the time we won’t rely on default user/password.**
* **We can provide own user name and password through application.properties file by adding security.basic.enabled=true, security.user.name=admin, security.user.password=root**

**27.cache mechanism/ehcache/what is cache tool usedf**

**28. request headers**

* **A request header is an**[**HTTP header**](https://developer.mozilla.org/en-US/docs/Glossary/header)**that can be used in an HTTP request to provide information about the request**

**Fields of request header-**

**1.Accept -** [**Media type(s)**](https://en.wikipedia.org/wiki/Media_type)**that is/are acceptable for the response. Eg: application/json**

**2. Accept-Language – acceptable languages:en-us**

**3. Access-Control-Request- Method,Access-Control-Request-Headers- Access-Control-Request-Method: GET**

**4.host - The domain name of the server (for**[**virtual hosting**](https://en.wikipedia.org/wiki/Virtual_hosting)**), and the**[**TCP port**](https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers)**number on which the server is listening. The**[**port**](https://en.wikipedia.org/wiki/Port_(computer_networking))**number may be omitted if the port is the standard port for the service requested.**

**Eg: 10.231.190.88:30323**

**5.origin - Initiates a request for**[**cross-origin resource sharing**](https://en.wikipedia.org/wiki/Cross-origin_resource_sharing)**(asks server for**[**Access-Control-\***](https://en.wikipedia.org/wiki/List_of_HTTP_header_fields#access-control-response-headers)**response fields).**

**Origin:** [**http://www.example-social-network.com**](http://www.example-social-network.com)

**6. Content-Type - The**[**MIME type**](https://en.wikipedia.org/wiki/MIME_type)**of this content - : application/json**

**29.spring boot components**

**30. what is spring cloud?**

**31. How to identify the service which is causing an issue in spring boot?**

**32. CORS in spring boot**

* **“CORS” stands for Cross-Origin Resource Sharing.**
* **It allows you to make requests from one website to another website in the browser, which is normally prohibited by another browser policy called the Same-Origin Policy (SOP).**
* **SOP means the resource can be accessed by the same origin. To overcome this we are using CORS.**
* **CORS providing control over the resource how it is accessed outside domain**

**33. what is interceptors?**

**You can use the Interceptor in Spring Boot to perform operations like writing the log, do some configuration under the following situations −**

* **Before sending the request to the controller – PreHandle()**
* **Before sending the response to the client – postHandle()**

**afterCompletion() – called after completing the request and response**

**For example, you can use an interceptor to add the request header before sending the request to the controller and add the response header before sending the response to the client.**

**34. what is filter in spring boot?**

**Filters as the name suggest used to perform filtering on either the request to a resource or on the response from a resource, or both. *Spring Boot* provides few options to *register custom filters* in the *Spring Boot application*.With the help of filter, we can perform the following operations.**

1. **Perform some request processing before the request is handed over to the controller.**
2. **Processing response before it reaches the client.**

**35.what is swagger**

* **swagger is used to document the Rest API webservices**
* **if we done any changes in it will automatically change in documentation**
* **we don’t need to manage manually.**

**Steps to create swagger**

* + - 1. **add swagger dependency in build tool**
      2. **create configuration file and annotate with @EnableSwagger2**
      3. **write a method where we have to mention the controller base package and it will generate the documentation**

**36. What is loosely coupling**

**Loosely coupling in java**

* **In simple words, loose coupling means they are mostly independent. If the only knowledge that class A has about class B, is what class B has exposed through its interface, then class A and class B are said to be loosely coupled**
* **Example: Imagine you have created two classes, A and B, in your program. If you change class A volume, then you are not forced to change class B. This is called loose coupling in Java.**
* **When class A requires changes in class B, then you have tight coupling.**

### **Tightly Coupling**

* **When two classes are highly dependent on each other, it is called tightly coupling.**
* **It occurs when a class takes too many responsibilities or where a change in one class requires changes in the other class.**
* **Example: Imagine you have created two classes A and B, in your program. Class A is called volume, and class B evaluates the volume of a cylinder. If you make any changes in the volume, then the same changes will reflect in class B. Hence, we can say both the classes are highly dependent on each other and are tightly coupled.**

**37. different between Query param and path param**

* **Path Param is basically used to identify a specific resource or resources whereas Query Parameter is used to sort/filter those resources.**
* **Hence Path Param is mandatory, query param is optional to pass value**
* **Let's consider an example where you want identify the employee on the basis of employeeID, and in that case, you will be using the URI param.**
* **GET /employee/{employeeID}**
* **Take another example where you want to filter the employee on the basis of designation, and in that case, you will be using Query Parameter.**
* **GET /employee?designation=SSE**

**Ex:**

**@RestController**

**@RequestMapping(path = "/spring-mvc-basics ")**

**public class PrecautionsController**

**{**

**@GetMapping("/foos/{id}")**

**public String getFooById(@PathVariable(“id”) String id) { return "ID: " + id; }**

**/\*\***

**Then we can map based on the path:**

**http://localhost:8080/spring-mvc-basics/foos/abc**

**----**

**ID: abc \*\*/**

**@GetMapping("/foos")**

**@ResponseBody**

**public String getFooByIdUsingQueryParam(@RequestParam String id) {**

**return "ID: " + id;**

**}**

**/\*\***

**which would give us the same response, just a different URI:**

**http://localhost:8080/spring-mvc-basics/foos?id=abc**

**----**

**ID: abc**

**\*\*/**

**what Is spring AOP?**

* **Spring AOP enables Aspect-Oriented Programming in spring applications.**
* **In AOP, aspects enable the modularization of concerns such as transaction management, logging or security that cut across multiple types and objects (often termed crosscutting concerns).**
* **It entails to break down application logic into different parts.**
* **AOP provides the way to dynamically add the cross-cutting concern before, after or around the actual logic using simple pluggable configurations. It makes easy to maintain code in the present and future as well.**
* **You can add/remove concerns without recompiling complete source code simply by changing configuration files (if you are applying aspects suing XML configuration).**

[**https://howtodoinjava.com/spring-aop-tutorial/**](https://howtodoinjava.com/spring-aop-tutorial/)

[**https://medium.com/@bushra.saifi/spring-aop-18ebdcf669a1**](https://medium.com/@bushra.saifi/spring-aop-18ebdcf669a1)

**how you will manage the transaction management in springboot with multiple database**

**bean factory and application context**

**how to get the list of employees, who has age is more than 25? (default method)**

**how to get the list of employees, by passing first name?(default method/customize method)**

**what is build.gradlifference between gradle and maven?**

**Microservices**

**Note : https://www.guru99.com/microservices-interview-questions.html**

**1.what is microservice architecture.**

* **Microservices is an architectural style that structures an application as a collection of services that are highly maintainable and testable, loosely couple, independently deployable and modeled around a business domain.**
* **So large applications using this architectural pattern can be broken into small multiple micro services.**
* **Which together act as a one large application. But behind the scene it’s a microservices. These microservices communicating each other through Api’s**

**2.what is advantage and disadvantage of microservice**

**Advantages:**

* **Each service can only focus on one single business capability.**
* **It is possible to change or upgrade each service individually rather than upgrading in the entire application.**
* **Less dependency, and easy to test. Faster release cycle.**

**Dis Advantages:**

* **Microservices has all the complexities of the distributed system.**
* **There is a higher chance of failure during communication between different services.**
* **Difficult to manage a large number of services.**
* **Complex testing because of a distributed environment.**
* **When more services interact with each other, the possibility of failure also increases.**

**3.how two microservice are communicating**

**4. Fault tolerance**

* **Fault tolerance is the individual service that does not bring down the overall system.**
* **Without fault tolerance, a single failure in the system may cause a total breakdown.**
* **Consider a scenario in which six microservices are communicating with each other. The microservice-5 becomes down at some point, and all the other microservices are directly or indirectly depend on it, so all other services also go down.**
* **The solution to this problem is to use a fallback in case of failure of a microservice. This is called fault tolerance.**
* **Fault tolerance can be achieved with the help of a circuit breaker.**
* **The circuit breaker is a pattern that wraps the request to external service and detects when they are faulty.**
* **If a failure is detected, the circuit breaker opens. All the subsequent requests immediately return an error instead of making requests to the unhealthy service.**

**5. what is Circuit Breaker**

* **Circuit breaker is used to identify which of the services is not running .**
* **If a failure is detected, the circuit breaker opens. All the subsequent requests immediately return an error instead of making requests to the unhealthy service.**
* **It rejects calls until it becomes healthy again**
* **Till now it will run the fall back method available**

**6. what is Euraka Server or Discovery Server**

* **Eureka Server is an application that holds the information about all client-service applications.**
* **Every Micro service will register into the Eureka server and Eureka server knows all the client applications running on each port and IP address.**
* **Eureka Server is also known as Discovery Server**
* **Whenever a service wants to talk with another service it should first talk with eureka naming server.**
* **The naming server provides the instances of particular service that are currently running.**
* **The process of providing instances to other services is called Service Discovery.**

**Steps:**

**1. Create Eureka Server**

* **Create spring boot application**
* **Add eureka server starter in build tool dependency**
* **Add EnableEurekaServer annotation in main class**

**2. Create Eureka Client**

* **Eureka discovery client started in build tool dependency**
* **Add EnableEurekaClient annotation in main class**
* **Add eureka service properties, url in application.properties file**
* **When call another microservice we don’t need to mention the ip address , we can simply specify the application name which is registered in Eureka**

**7. what is API Gateway**

* **The API Gateway is a server. It is a single entry point into a system. API Gateway encapsulates the internal system architecture**
* **All the requests made by the client go through the API Gateway. After that, the API Gateway routes requests to the appropriate microservice.**
* **It also has an inbuilt load balancer to load the balance of all incoming request from the client.**
* **Default port: 8765**

**There is a total of two different methods by which the API gateway handles the request made by the client:**

* **It routes the request of the client to a suitable service.**
* **It spreads out the request of clients to multiple services.**

**Advantage:**

* **The most important advantage of API Gateway is that it encapsulates the internal structure of the application.**
* **Rather than invoking the specific service, the client directly talks to the API Gateway.**

**Dis Advantage:**

* **It requires routing rules.**
* **There is a possibility of a single point of failure.**
* **Risk of complexity due to all the API rules are in one place.**

**8. hystrix**

* **Hystrix is a library. Using this we can implement the dashboard there we can identify which are the services running and which are not running.**

**Steps :**

* **Add EnableHystrix annotation in API Gateway main class**
* **Add hystrix dependency in build tool**

**9. what is distributed Tracing?**

* **We use distributed tracing to follow the request or transaction as it travels through the application.**
* **This is use to pinpoint bugs, or other issues that impact the application’s performance.**
* **We can use Zipkin. (We use new Relic)**

**10.what is Load balancer**

* **Load balancing is nothing but efficient distribution of network or application traffic across multiple servers.**
* **Each load balancer sits between client devices and backend servers, receiving and then distributing incoming requests to any available server capable of fulfilling them.**
* **We can use netflix ribbon as a load balancer. It provides the client-side balancing algorithm. It uses a Round Robin Load Balancing**

**There are two types of load balancing available:**

* **Server Side Load Balancing: Server side load balancing is a monolithic It applies between the client and the server. It accepts incoming network, application traffic, and distributes the traffic across the multiple backend servers by using various methods. The middle component is responsible for distributing the client requests to the server.**
* **Client-Side Load Balancing: The client holds the list of server’s IPs so that it can deliver the requests. The client selects an IP from the list, randomly, and forwards the request to the server.**

**10. web client**

**11. rest template**

**12. Feign client**

**13. difference b/w web client and feign client**

**14. Orchestration in microservice**

**.**

**2.stream all methods and programs, reverse programs, count the number of car in string, all other interv programs , lambda expression example program**

**3. exectur framework, Async process,Multi treadin, fpa thread scheduler,Future,CompletableFuture, ,**

**4. Desin pattern - singleton, factory , abstract factory pattern, template pattern , facade pattern**

**6.sprin cloud, AOP,**

**7. how to handle the failure of communication (circuit breaker etc)**

**8. all intervw sql query , sample procedure and procedure**

**Interview question**

**AOP**

**Spring Batch processingo**

**Spring security – Oauth , JWT**

**Spring Scheduling**

**Messaging System**

**How you will handle validations in Spring boot**

**How to handle the exception in spring boot**

**How to stream the List<List<String>list; -flatmap syntax**

**constructor injection vs setter injection**

**@autowired vs @injects**

**Write a program for below scenario**

**input=[1,2,3,4,5,6]**

**1) groupSize=4 and iterations=5**

**output:**

**1234**

**5612**

**3456**

**1234**

**5612**

**2) groupSize=5 and iterations = 2**

**output 12345**

**61234**

**Write the program to compare two employees if emp first name equal then comp employee last name , if it also equal then find employee city**

**git commands**

**rebase**

**switch**

**cascadeTyle-ALL**

**@requeriednew**

**Diff b/w @Component and @service**

**Immutable Design Pattern**